Codificação de Canal

- Quando informação digital é enviada através de um canal de transmissão, ruído e interferência inerentes a qualquer canal prático degradam o sinal de forma que os dados recebidos contêm erros.
- O usuário do sistema de transmissão digital geralmente estabelece uma taxa de erro máxima aceitável – uma mensagem errada em 1×10^{6} mensagens recebidas, por exemplo (i.e., uma taxa de erro de 1×10^{-6}) – acima da qual os dados recebidos não são considerados utilizáveis pelo usuário. Esta taxa de erro máxima aceitável depende da informação que transita pelo canal.
- A título de comparação, a taxa máxima de erro permitida para transmissão de voz através de telefonia celular é muito maior do que a taxa exigida para transmissão de dados, por exemplo. Até porque, na pior das hipóteses, mesmo sob uma alta taxa de erro e conseqüente distorção, o sistema auditivo humano é capaz de compreender o significado das frases pelo contexto da conversa, o que já não acontece quando dois computadores trocam dados.

O Codificador de Canal é o responsável em um sistema digital por manter a taxa de erro dentro de um limite máximo aceitável pelo usuário.

A possibilidade do uso de codificação para controlar com eficiência a taxa de erro de um sistema de comunicação digital foi demonstrada por Shannon [Shannon] em 1948 através do denominado **Teorema Fundamental de Shannon**:

> Se a taxa (= velocidade) de transmissão R $[bits/s]$ da informação a ser enviada pelo canal é menor que uma quantidade C $[bits/s]$ denominada de Capacidade do Canal, então a comunicação através do canal pode ser estabelecida com uma probabilidade de erro tão baixa quanto se deseje através do uso de um código adequado para correção de erro.

Em essência, o Teorema Fundamental de Shannon estabelece que a potência do sinal transmitido, a potência de ruído no canal e a largura de banda do canal estabelecem um limite máximo na taxa de transmissão R.

No caso específico de o único agente degradante do canal ser ruído $\eta(t)$ com distribuição de probabilidade Gaussiana (canal Gaussiano), a **Lei de Shannon-Hartley**, decorrente do Teorema Fundamental de Shannon, estabelece que a capacidade *C* deste tipo de canal é dada por

$$C = B\log_2\left(1+\frac{P}{N}\right)\ [bits/s] \quad (4.1)$$ onde:

B é a largura de banda do canal em Hz,
P é a potência do sinal transmitido e
N é a potência do ruído Gaussiano adicionado ao sinal no canal.

Outra interpretação é a de que *P* é a potência do sinal recebido no receptor e *N* é a potência do ruído na entrada do receptor.

A Lei de Shannon-Hartley apresenta duas importantes implicações:

1- Ela dá um limite superior para a velocidade (taxa) de transmissão confiável através de um canal Gaussiano.

2- Para uma Capacidade de Canal *C* especificada, ela define o compromisso entre a largura de banda *B* do canal e a relação sinal-ruído $\text{SNR} = P/N$ (SNR – *Signal To Noise Ratio*) no mesmo.

Apesar de (4.1) somente ser válida para canal AWGN (AWGN – *Additive White Gaussian Noise*), isto é, o ruído aditivo $\eta(t)$ do canal é Gaussiano e descorrelacionado (i.e., espectralmente branco - *white*), a Lei de Shannon-Hartley é de utilidade prática por duas razões:

1- Em geral a maioria dos canais físicos são pelo menos aproximadamente AWGN.

2- Demonstra-se que o resultado obtido para um canal AWGN provê um limite inferior para a performance de um sistema digital operando com um canal não AWGN.

Em geral, como a densidade espectral de $\eta(t)$, dada por $|\Im\{\eta(t)\}|^2$, é uma constante $\eta_0/2$ dentro do intervalo de freqüência $-B \le f \le B$, o ruído pode ser considerado ruído branco [Taub] e a potência do ruído pode ser aproximada por $N = \eta_0 B$, sendo $\Im\{\cdot\}$ o operador Transformada de Fourier [Carlson], e (4.1) pode ser reescrita como $C = B\log_2\left(1+\frac{P}{\eta_0 B}\right)[bits/s]$.

Limite de Shannon

Quando a largura de banda *B* do canal é aumentada ao infinito, a Capacidade do Canal resulta em

$$C\big|_{B\to\infty} = \frac{P}{\eta_0}\log_2 e = 1.44\frac{P}{\eta_0}\ \left[bits/s\right] \qquad (4.6)$$

A Equação (4.6) é conhecida como Limite de Shannon.

O Limite de Shannon define a máxima taxa de transmissão para um canal cuja largura de banda seja suficientemente grande, tal que não apresente qualquer atenuação ao espectro do sinal que transporta a informação a ser transmitida.

- Infelizmente, o Teorema Fundamental de Shannon apenas demonstra que se $R \le C$ existe um código corretor de erro tal que a informação pode ser transmitida através do canal com uma taxa de erro arbitrariamente baixa, mas não especifica como construir tal código corretor.
- Talvez a maior utilidade prática do Teorema Fundamental de Shannon seja demonstrar que para $R > C$ não é possível transmitir informação sem erro através do canal, mesmo que se utilize o mais poderoso código corretor de erro que se possa conceber.
- É importante salientar que, não raro, o maior valor possível para a taxa de transmissão *R* é dado não por *C*, mas sim, pela complexidade computacional do código corretor necessário para que aquele valor de *R* possa ser alcançado.

4.1 Códigos Corretores de Erro

Vimos que o Teorema Fundamental de Shannon estabelece a existência de um código corretor de erro tal que a informação pode ser transmitida através do canal de comunicação com uma taxa de erro arbitrariamente baixa, caso a taxa de transmissão R $[bits/s]$ seja menor ou igual à capacidade do canal C $[bits/s]$.

Estudaremos agora como construir tais códigos.

Especificamente, estudaremos os membros mais importantes de duas grandes classes de códigos para correção de erro: os códigos de bloco e os códigos convolucionais.

É importante lembrar que o processo de correção de erros através de codificação/decodificação é realizado no Codificador/Decodificador de Canal.

Algumas vezes este processo é referido como:

FEC (FEC – *Forward Error Correction*) = procura inferir e imediatamente corrigir erros pelas características do sinal recebido;

ARQ (ARQ – *Automatic Repeat Request*) = simplesmente detecta a existência de erros no receptor e solicita ao transmissor que envie a informação original novamente. Isto implica na existência de um canal de *feedback* do receptor para o transmissor, de modo que o processo ARQ não é muito utilizado exceto quando o objetivo são baixíssimas taxas de erro [Taub].

4.2 Códigos de Bloco

Podemos considerar um código de bloco de maneira semelhante àquela que adotamos para os códigos compressores por entropia vistos no Capítulo III.

Ou seja, um código de bloco pode ser considerado como um operador $\boldsymbol{\theta}\{\cdot\}$, tal que $\mathbf{C} = \boldsymbol{\theta}\{\mathbf{X}\}$, onde $\mathbf{X} = \{\underline{x}_i\} = \{\underline{x}_0, \underline{x}_1, \cdots, \underline{x}_{M-1}\}$ é o conjunto de M possíveis **mensagens** $\underline{x}_i$ a serem codificadas e $\mathbf{C} = \{\underline{c}_i\} = \{\underline{c}_0, \underline{c}_1, \cdots, \underline{c}_{M-1}\}$ é o conjunto de M possíveis **palavras-código** $\underline{c}_i$ resultantes da codificação, com $i = 0,1,\cdots,M-1$.

O operador $\boldsymbol{\theta}\{\cdot\}$ efetua um mapeamento unívoco entre cada mensagem $\underline{x}_i$ e a respectiva palavra-código $\underline{c}_i$.

O **conjunto de caracteres do código** ou **alfabeto do código** é o conjunto $\mathbf{A} = \{a_0, a_1, \cdots, a_{D-1}\}$ composto por D elementos, de cuja composição são formadas cada mensagem e sua respectiva palavra-código.

No contexto de códigos corretores de erro:

- Cada mensagem $\underline{x}_i \in \mathbf{X}$ é considerada como um vetor $\underline{x}_i = [x_{i(k-1)} \quad x_{i(k-2)} \quad \cdots \quad x_{i1} \quad x_{i0}]$ de k componentes $x_{ij} \in \mathbf{A}$, $j = k-1, k-2, \cdots, 1, 0$.
- Visto que os k componentes da i-ésima mensagem $\underline{x}_i$ pertencem ao alfabeto **A**, é válida a relação de pertinência $\underline{x}_i \in \mathbf{A}^k$.
- Da mesma forma, cada palavra-código $\underline{c}_i \in \mathbf{C}$ é considerada como um vetor $\underline{c}_i = [c_{i(n-1)} \quad c_{i(n-2)} \quad \cdots \quad c_{i1} \quad c_{i0}]$ de n componentes $c_{ij} \in \mathbf{A}$, $j = n-1, n-2, \cdots, 1, 0$.
- Visto que os n componentes da i-ésima palavra-código $\underline{c}_i$ pertencem ao alfabeto **A**, é válida a relação de pertinência $\underline{c}_i \in \mathbf{A}^n$.

Por exemplo, a palavra-código binária 0101, de $n = 4$ bits, é representada pelo vetor $\underline{c} = [0 \quad 1 \quad 0 \quad 1]$, $\underline{c} \in \mathbf{A}^4$, $\mathbf{A} = \{0,1\}$.

⇒ Para um código D – ário, com D sendo uma potência inteira de 2 (i.e. $D = 2^b$, onde b é um inteiro positivo), cada caractere D – ário terá uma representação binária equivalente formada por uma seqüência de b bits.

⇒ Portanto, um tal código D – ário cujo tamanho da palavra-código é de N caracteres D – ários pode ser mapeado em um código binário cujo tamanho da palavra-código é $n = bN$.

⇒ Como, em geral, $D = 2^b$ em sistemas práticos, neste estudo focalizaremos os códigos binários ($\mathbf{A} = \{0,1\}$), visto que a qualquer instante o código D – ário pode ser mapeado no código binário equivalente e vice-versa.

Exemplo 4.1: Determine o código binário equivalente ao código $\boldsymbol{\theta}\{\cdot\}$ abaixo.

Mensagem $\underline{x}_i$	Palavra-código $\underline{c}_i$ associada a $\underline{x}_i$ por $\underline{c}_i = \boldsymbol{\theta}\{\underline{x}_i\}$
00	00
01	03
02	11
03	12
10	21
11	22
12	30
13	33

Solução:

O código original é quaternário ($D = 4$).

O número de caracteres quaternários utilizado na representação das palavras-código do código quaternário é $N = 2$.

Existem $M = 8$ mensagens quaternárias, logo o código binário equivalente necessita $k = \log_2 M = 3$ bits em cada mensagem binária para representá-las.

Assim, ao mapear o conjunto de M mensagens quaternárias no conjunto de M mensagens binárias, obtemos:		O número de bits necessários a cada palavra-código binária equivalente é $n = bN$, onde $b = \log_2 D = 2$. Logo $n = bN = 2 \times 2 = 4$. Assim, ao mapear o conjunto de M palavras-código quaternárias no conjunto de M palavras-código binárias, obtemos:		Portanto o código binário equivalente ao código quaternário é (Tab. 4.1):	
Mensagem Quaternária	Mensagem Binária Equivalente de $k = 3$ bits	Palavra-Código Quaternária	Palavra-Código Binária Equivalente de $n = 4$ bits	Mensagem $\underline{x}_i$ de $k = 3$ bits	Palavra-código $\underline{c}_i$ de $n = 4$ bits associada a $\underline{x}_i$ por $\underline{c}_i = \boldsymbol{\theta}\{\underline{x}_i\}$.
00	000	00	0000	000	0000
01	001	03	0011	001	0011
02	010	11	0101	010	0101
03	011	12	0110	011	0110
10	100	21	1001	100	1001
11	101	22	1010	101	1010
12	110	30	1100	110	1100
13	111	33	1111	111	1111

4.3 Códigos de Bloco Binários

$\Rightarrow$ Um código de bloco binário $\boldsymbol{\theta}\{\cdot\}$ mapeia um conjunto $\mathbf{X} = \{\underline{x}_i\} = \{\underline{x}_0, \underline{x}_1, \cdots, \underline{x}_{M-1}\}$ de $M = 2^k$ mensagens binárias, cada uma delas com *k* bits, em um conjunto $\mathbf{C} = \{\underline{c}_i\} = \{\underline{c}_0, \underline{c}_1, \cdots, \underline{c}_{M-1}\}$ de M palavras-código binárias, cada uma delas com *n* bits, onde $n > k$.

$\Rightarrow$ Um código de bloco $\boldsymbol{\theta}\{\cdot\}$ binário cujas mensagens a serem codificadas apresentam *k* bits e são mapeadas em palavras-código de *n* bits é representado pelo operador $\boldsymbol{\theta}(n,k)\{\cdot\}$ ou simplesmente $\boldsymbol{\theta}(n,k)$.

- Um código $\boldsymbol{\theta}(n,k)$ é **sistemático** quando cada palavra-código de *n* bits é formada pelos *k* bits da respectiva mensagem associada, acrescidos (por justaposição) de *r* bits adicionais destinados ao controle e correção de erros, denominados de **bits de paridade**.
- Portanto, em um código sistemático cada mensagem contendo *k* bits de informação é expandida em uma palavra-código de $n = k + r$ bits onde *r* é o número de bits representativos da informação redundante adicionada visando o controle e correção de erro.
- Um código $\boldsymbol{\theta}(n,k)$ é **não-sistemático** quando nas palavras-códigos de *n* bits não aparecem explicitamente representados os *k* bits de informação da respectiva mensagem associada.
- Na Seção 4.3.2 veremos como converter um código não-sistemático em um código sistemático. Em função disto, inicialmente restringiremos nossa atenção aos códigos sistemáticos.

Por exemplo, o código $\boldsymbol{\theta}(4,3)$ da Tabela 4.1 é sistemático porque cada palavra-código $\underline{c}_i$ de $n = 4$ bits é formada pela justaposição de 1 bit de paridade aos $k = 3$ bits de informação da mensagem $\underline{x}_i$ associada.

Observe que, como $n > k$, no conjunto de todas as 2^n possíveis palavras-códigos de *n* bits existem $2^n - 2^k$ elementos que não são associados a qualquer elemento do conjunto $\mathbf{X} = \{\underline{x}_i\} = \{\underline{x}_0, \underline{x}_1, \cdots, \underline{x}_{M-1}\}$ de $M = 2^k$ mensagens binárias de *k* bits.

Por exemplo, para o código binário $\boldsymbol{\theta}(4,3)$ da Tabela 4.1 existem $2^n - 2^k = 2^4 - 2^3 = 8$ elementos no conjunto de todas as $2^n = 2^4 = 16$ possíveis palavras-códigos de 4 bits sem associação com qualquer mensagem do conjunto $\mathbf{X} = \{000, 001, 010, 011, 100, 101, 110, 111\}$.

⇒ Em geral é desejável que o tempo $n\tau_s$ de duração de uma palavra-código seja igual (idealmente menor ou igual) ao tempo de duração $k\tau_x$ de uma mensagem, onde τ_s representa a largura (duração) dos bits em uma palavra-código e τ_x representa a largura dos bits em uma mensagem.

⇒ Assim, se $n\tau_s = k\tau_x$, então a razão de codificação R_c de um código de bloco é $R_c = k/n = \tau_s/\tau_x$.

- O **peso** de uma palavra-código é definido como o número de dígitos "1" nela presentes.
- O conjunto de todos os pesos de um código constitui a **distribuição de pesos** do código.
- Quando todas as *M* palavras-código têm pesos iguais, o código é denominado de **código de peso constante**.
- Por exemplo, o peso da palavra-código $\underline{c} = [0 \quad 1 \quad 0 \quad 1]$ é 2.

O processo de codificação/decodificação de um código de bloco baseia-se na propriedade algébrica de que o conjunto de **palavras-código** $\mathbf{C} = \{\underline{c}_i\} = \{\underline{c}_0, \underline{c}_1, \cdots, \underline{c}_{M-1}\}$ pode ser mapeado em um conjunto de polinômios $\{C_i(p)\} = \{C_0(p), C_1(p), \cdots, C_{M-1}(p)\}$.

→ Os componentes do **vetor** $\underline{c}_i = \begin{bmatrix} c_{i(n-1)} & c_{i(n-2)} & \cdots & c_{i1} & c_{i0} \end{bmatrix}$ que representa a *i*-ésima **palavra-código** correspondem aos coeficientes do polinômio $C_i(p) = c_{i(n-1)} p^{n-1} + c_{i(n-2)} p^{n-2} + \cdots + c_{i1} p + c_{i0}$ associado à palavra-código.

A mesma propriedade algébrica pode ser aplicada sobre o conjunto de **mensagens** $\mathbf{X} = \{\underline{x}_i\} = \{\underline{x}_0, \underline{x}_1, \cdots, \underline{x}_{M-1}\}$ de modo que este também pode mapeado em um conjunto de polinômios $\{X_i(p)\} = \{X_0(p), X_1(p), \cdots, X_{M-1}(p)\}$.

→ Os componentes do **vetor** $\underline{x}_i = \begin{bmatrix} x_{i(k-1)} & x_{i(k-2)} & \cdots & x_{i1} & x_{i0} \end{bmatrix}$ que representa a *i*-ésima **mensagem** correspondem aos coeficientes do polinômio $X_i(p) = X_{i(k-1)} p^{n-1} + c_{i(k-2)} p^{n-2} + \cdots + c_{i1} p + c_{i0}$ associado à mensagem.

Por este motivo os códigos de bloco são também denominados de **códigos polinomiais**.

Por exemplo, a representação polinomial do código da Tabela 4.1 é mostrada na Tabela 4.2.

Tabela 4.2: Representação polinomial do código da Tabela 4.1.

Mensagem $\underline{x}_i$	Polinômio $X_i(p)$	Palavra-código $\underline{c}_i$ associada a $\underline{x}_i$ por $\underline{c}_i = \boldsymbol{\theta}\{\underline{x}_i\}$	Polinômio $C_i(p)$
000	0	0000	0
001	1	0011	$p+1$
010	p	0101	p^2+1
011	$p+1$	0110	p^2+p
100	p^2	1001	p^3+1
101	p^2+1	1010	p^3+p
110	p^2+p	1100	p^3+p^2
111	p^2+p+1	1111	p^3+p^2+p+1

- O processo de codificação/decodificação envolve operações aritméticas de adição e multiplicação realizadas sobre o conjunto de polinômios $\{C_i(p)\} = \{C_0(p), C_1(p), \cdots, C_{M-1}(p)\}$ que representam as palavras-código, conforme veremos.
- Um código corretor de erro deve ser tal que o conjunto $\{C_i(p)\}$ e as operações aritméticas sobre ele definidas obedeçam a determinadas restrições, caso contrário a unicidade e o custo computacional do processo de codificação/decodificação resultarão prejudicados.
- Especificamente, os coeficientes dos polinômios em $\{C_i(p)\}$ devem pertencer a um tipo especial de conjunto denominado de **campo algébrico** (*field*) [Chen].
- Um campo algébrico é uma entidade matemática estudada em Álgebra Linear.

Campo Algébrico

Um campo **F** é um conjunto de elementos que permite duas operações sobre seus elementos – adição e multiplicação – e que satisfaz aos seguintes axiomas (propriedades):

Adição

1- O conjunto $\mathbf{F}$ é fechado sob adição, i.e., se $a,b \in \mathbf{F}$ então $a+b \in \mathbf{F}$.

2- A adição em $\mathbf{F}$ é associativa, i.e., se $a,b,c \in \mathbf{F}$ então $a+(b+c)=(a+b)+c$.

3- A adição em $\mathbf{F}$ é comutativa, i.e., se $a,b \in \mathbf{F}$ então $a+b=b+a$.

4- O conjunto $\mathbf{F}$ contém um único elemento denominado **zero**, representado por "0", que satisfaz a condição $a+0=a$, $\forall a \in \mathbf{F}$.

5- Cada elemento em $\mathbf{F}$ tem o seu elemento negativo (simétrico). Se $b \in \mathbf{F}$ então seu simétrico é denotado por $-b$ tal que $b+(-b)=0$. Se $a \in \mathbf{F}$ então a subtração $a-b$ entre os elementos *a* e *b* é definida como $a+(-b)$.

Multiplicação

1- O conjunto $\mathbf{F}$ é fechado sob multiplicação, i.e., se $a,b \in \mathbf{F}$ então $ab \in \mathbf{F}$.

2- A multiplicação em $\mathbf{F}$ é associativa, i.e., se $a,b,c \in \mathbf{F}$ então $a(bc)=(ab)c$.

3- A multiplicação em $\mathbf{F}$ é comutativa, i.e., se $a,b \in \mathbf{F}$ então $ab=ba$.

4- A multiplicação em $\mathbf{F}$ é distributiva sobre a adição, i.e., se $a,b,c \in \mathbf{F}$ então $a(b+c)=ab+ac$.

5- O conjunto $\mathbf{F}$ contém um único elemento denominado **identidade**, representado por "1", que satisfaz a condição $1a=a$, $\forall a \in \mathbf{F}$.

6- Cada elemento de $\mathbf{F}$, exceto o elemento 0, possui um elemento **inverso**. Assim, se $b \in \mathbf{F}$ e $b \neq 0$ então o inverso de *b* é definido como b^{-1} tal que $bb^{-1}=1$. Se $a \in \mathbf{F}$ então a divisão $a-b$ entre os elementos *a* e *b* é definida como ab^{-1}.

Por exemplo, o conjunto $\Re$ dos números reais é um campo algébrico com infinitos elementos, assim como também o é conjunto dos números complexos $\mathbb{C}$. Estes dois conjuntos obedecem aos axiomas acima.

- Um campo algébrico finito com D elementos é denominado de Campo de Galois (*Galois Field*) e é designado por $\mathbf{GF}(D)$.
- Nem para todos os valores de D é possível formar um campo.
- Em geral, quando D é primo (ou uma potência inteira de um número primo) é possível construir o campo finito $\mathbf{GF}(D)$ consistindo dos elementos $\{0,1,\cdots,D-1\}$, desde que as operações de adição e multiplicação sobre $\mathbf{GF}(D)$ sejam operações **módulo *D*** [Clark].
- **Nota**: Uma operação **op** é módulo D quando pode ser representada por $(a\ \mathbf{op}\ b)\bmod D$, onde $x \bmod y$ é o operador que resulta no resto da divisão x/y.
- Por exemplo, a operação de soma módulo 5 entre os números 4 e 3 resulta em 2 visto que $(4+3)\bmod 5 = 2$.

Por exemplo, o Campo de Galois $\mathbf{GF}(2)$ é formado pelo conjunto $\{0,1\}$ e pelas operações módulo 2 de soma e multiplicação dadas pelas Tabelas 4.3 e 4.4

Tabela 4.3: Soma sobre $\mathbf{GF}(2)$

+	0	1
0	0	1
1	1	0

Tabela 4.4: Multiplicação sobre $\mathbf{GF}(2)$

.	0	1
0	0	0
1	0	1

Note nas Tabelas 4.3 e 4.4 que:

$\Rightarrow$ a soma entre dois elementos *a* e *b* pertencentes a $\mathbf{GF}(2)$ é implementada pela

operação lógica $a \oplus b$ (ou a XOR b) e que

$\Rightarrow$ a multiplicação entre dois elementos *a* e *b* pertencentes a $\mathbf{GF}(2)$ é implementada pela

operação lógica $a.b$ (ou a AND b).

Por isto é usual os códigos corretores serem construídos em $\mathbf{GF}(2)$ dada a facilidade de implementação com portas lógicas AND e XOR.

Assim, um código corretor de erro binário ($\mathbf{A} = \{0,1\}$) é tal que os coeficientes dos polinômios em $\{C_i(p)\}$ pertencem a $\mathbf{GF}(2) = \mathbf{A} = \{0,1\}$ e as operações aritméticas realizadas sobre o conjunto de polinômios $\{C_i(p)\} = \{C_0(p), C_1(p), \cdots, C_{M-1}(p)\}$ (ou, equivalentemente, sobre o conjunto de palavras-código $\mathbf{C} = \{\underline{c}_i\} = \{\underline{c}_0, \underline{c}_1, \cdots, \underline{c}_{M-1}\}$) durante o processo de codificação/decodificação obedecem às Tabelas 4.3 e 4.4.

4.3.1 Capacidade de Detecção e Correção de Erro

⇒ Suponhamos que $\underline{c}_i$ e $\underline{c}_j$ sejam duas palavras-código quaisquer do código $\boldsymbol{\theta}(n,k)$.

⇒ Uma medida da diferença entre as duas palavras-código é o número de bits em posições correspondentes que diferem entre si.

⇒ Esta medida é denominada de **Distância de Hamming** e é denotada por d_{ij}.

⇒ Por exemplo, sejam $\underline{c}_i = [0 \quad 1 \quad 0 \quad 1]$ e $\underline{c}_j = [1 \quad 0 \quad 0 \quad 0]$. Então $d_{ij} = 3$.

- Observe que d_{ij} sempre satisfaz a condição $0 < d_{ij} \leq n$, $i \neq j$, para duas palavras-código $\underline{c}_i$ e $\underline{c}_j$, ambas de n bits (por definição, em um código $\boldsymbol{\theta}(n,k)$, $\underline{c}_i \neq \underline{c}_j \ \forall i$ e $\forall j$ com $i \neq j$).
- O menor valor no conjunto $\{d_{ij}\}$, $i, j = 0,1,\cdots,M-1$, $i \neq j$, $M = 2^k$ é denominado **distância mínima** do código e é denotado como $d_{\min}$. Por exemplo, $d_{\min} = 2$ para o código $\boldsymbol{\theta}(4,3)$ da Tabela 4.1

- A Distância de Hamming d_{ij} é uma medida do grau de separação entre duas palavras-código $\underline{c}_i$ e $\underline{c}_j$.
- Assim, o grau de correlação temporal entre dois sinais modulados $v_i(t)$ e $v_j(t)$ gerados no modulador de um transmissor digital em decorrência de $\underline{c}_i$ e $\underline{c}_j$ é implicitamente associado à d_{ij} [Proakis].
- Portanto, $d_{\min}$ está associado à capacidade do código $\boldsymbol{\theta}(n,k)$ em identificar palavras-código demoduladas no receptor quando estas são recebidas em erro, como conseqüência do ruído e interferência presentes no canal.
- Em outras palavras, **quanto maior $d_{\min}$ maior a capacidade de um código $\boldsymbol{\theta}(n,k)$ detectar e corrigir erros.**

Demonstra-se que [Ash][Proakis]:

Seja $\boldsymbol{\theta}(n,k)$ um código corretor binário;

seja d o número máximo de erros que $\boldsymbol{\theta}(n,k)$ é capaz de **detetar;**

seja t o número máximo de erros que $\boldsymbol{\theta}(n,k)$ é capaz de **corrigir;**

seja $d_{\min}$ a distância mínima de $\boldsymbol{\theta}(n,k)$;

então:

$$d = d_{\min} - 1 \qquad (4.7)$$

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \qquad (4.8)$$

$$d_{\min} \leq n - k + 1 \qquad (4.9)$$

sendo $\lfloor . \rfloor$ o operador que resulta no inteiro mais próximo e menor que o argumento.

Por exemplo, $d_{\min} = 2$ para o código $\boldsymbol{\theta}(4,3)$ da Tabela 4.1.

Daí, de (4.7) e (4.8), temos que

$d = d_{\min} - 1 = 2 - 1 = 1$ e

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{2 - 1}{2} \right\rfloor = 0 .$$

Portanto o código $\boldsymbol{\theta}(4,3)$ da Tabela 4.1 detecta no máximo 1 erro por palavra-código mas não tem capacidade de correção. De fato, este código é um simples código *parity-check*.

4.3.2 A Matriz Geradora de um Código $\boldsymbol{\theta}(n,k)$

- Seja a *i*-ésima mensagem de um código binário $\boldsymbol{\theta}(n,k)$ representada pelo vetor $\underline{x}_i = \begin{bmatrix} x_{i0} & x_{i1} & \cdots & x_{i(k-1)} \end{bmatrix}$ e seja a *i*-ésima palavra-código de $\boldsymbol{\theta}(n,k)$ representada pelo vetor $\underline{c}_i = \begin{bmatrix} c_{i0} & c_{i1} & \cdots & c_{i(n-1)} \end{bmatrix}$, onde $i = 0,1,\cdots,M-1$, $M = 2^k$.

- O processo de codificação da mensagem $\underline{x}_i = \begin{bmatrix} x_{i0} & x_{i1} & \cdots & x_{i(k-1)} \end{bmatrix}$ na respectiva palavra-código $\underline{c}_i = \begin{bmatrix} c_{i0} & c_{i1} & \cdots & c_{i(n-1)} \end{bmatrix}$ efetuado por um código binário $\boldsymbol{\theta}(n,k)$ pode ser representado em forma matricial por

$$\underline{c}_i = \underline{x}_i \mathbf{G} \qquad (4.10)$$

onde a matriz $\mathbf{G}_{k\times n}$ é denominada de **matriz geradora** do código $\boldsymbol{\theta}(n,k)$ e é dada por

$$\mathbf{G} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0(n-1)} \\ g_{10} & g_{11} & \cdots & g_{1(n-1)} \\ \vdots & \vdots & & \vdots \\ g_{(k-1)0} & g_{(k-1)1} & \cdots & g_{(k-1)(n-1)} \end{bmatrix}. \qquad (4.11)$$

- Podemos interpretar a matriz **G** como um conjunto de k vetores-linha $\underline{g}_j$, $j = 0,1,\cdots,k-1$, tal que

$$\mathbf{G} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0(n-1)} \\ g_{10} & g_{11} & \cdots & g_{1(n-1)} \\ \vdots & \vdots & & \vdots \\ g_{(k-1)0} & g_{(k-1)1} & \cdots & g_{(k-1)(n-1)} \end{bmatrix} = \begin{bmatrix} \leftarrow & \underline{g}_0 & \rightarrow \\ \leftarrow & \underline{g}_1 & \rightarrow \\ & \vdots & \\ \leftarrow & \underline{g}_{(k-1)} & \rightarrow \end{bmatrix}. \qquad (4.12)$$

- Desta maneira, de $\underline{c}_i = \underline{x}_i \mathbf{G}$ (4.10) e (4.12), cada palavra-código $\underline{c}_i = \begin{bmatrix} c_{i0} & c_{i1} & \cdots & c_{i(n-1)} \end{bmatrix}$ é simplesmente uma combinação linear dos vetores $\underline{g}_j$ com coeficientes da combinação determinados pela mensagem associada $\underline{x}_i = \begin{bmatrix} x_{i0} & x_{i1} & \cdots & x_{i(k-1)} \end{bmatrix}$, isto é:

$$\underline{c}_i = x_{i0}\underline{g}_0 + x_{i1}\underline{g}_1 + \cdots + x_{i(k-1)}\underline{g}_{(k-1)} \qquad (4.13)$$

→ É possível demonstrar que [Clark][Peterson][Costello], o conjunto **C** de 2^k palavras-código de um código $\boldsymbol{\theta}(n,k)$ é um sub-espaço vetorial de dimensão k.

→ Logo, os k vetores-linha $\underline{g}_j$ que formam a matriz **G** devem ser linearmente independentes para que possam, conforme estabelece (4.13), gerar o sub-espaço **C** em k dimensões.

→ Em outras palavras, o conjunto de vetores $\underline{g}_j$ é uma base para o sub-espaço **C** .

Exemplo 4.2:

Verifique se a matriz $\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ é a matriz geradora do código $\boldsymbol{\theta}(4,3)$ da Tabela 4.1.

Solução: Cada palavra-código $\underline{c}_i = [c_{i0} \quad c_{i1} \quad \cdots \quad c_{i(n-1)}]$ de $n = 4$ bits é gerada através de (4.10) a partir da respectiva mensagem $\underline{x}_i = [x_{i0} \quad x_{i1} \quad \cdots \quad x_{i(k-1)}]$ de $k = 3$ bits. No total, existem $2^k = 8$ palavras-código em $\boldsymbol{\theta}(4,3)$. Assim,

$\underline{x}_i$	$\underline{x}_i\mathbf{G} = \underline{c}_i$		$\underline{x}_i$	$\underline{x}_i\mathbf{G} = \underline{c}_i$
$[0\ 0\ 0]$	$[0\ 0\ 0]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0\ 0\ 0\ 0]$		$[1\ 0\ 0]$	$[1\ 0\ 0]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [1\ 0\ 0\ 1]$
$[0\ 0\ 1]$	$[0\ 0\ 1]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0\ 0\ 1\ 1]$		$[1\ 0\ 1]$	$[1\ 0\ 1]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [1\ 0\ 1\ 0]$
$[0\ 1\ 0]$	$[0\ 1\ 0]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0\ 1\ 0\ 1]$		$[1\ 1\ 0]$	$[1\ 1\ 0]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [1\ 1\ 0\ 0]$
$[0\ 1\ 1]$	$[0\ 1\ 1]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0\ 1\ 1\ 0]$		$[1\ 1\ 1]$	$[1\ 1\ 1]\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [1\ 1\ 1\ 1]$

Portanto **G** é geradora de $\boldsymbol{\theta}(4,3)$.

Qualquer matriz geradora **G** de um código $\boldsymbol{\theta}(n,k)$ pode, através de operações elementares em suas linhas e permutações em suas colunas, ser reduzida à **forma sistemática** quando, então, o código gerado é sistemático.

Uma matriz geradora **G** encontra-se na forma sistemática quando

$$\mathbf{G} = \left[\mathbf{I}_k \;\vdots\; \mathbf{P}\right] = \left[\begin{array}{ccccc:ccc} 1 & 0 & 0 & \cdots & 0 & p_{00} & \cdots & p_{0(n-} \\ 0 & 1 & 0 & \cdots & 0 & p_{10} & \cdots & p_{1(n-} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{(k-1)0} & \cdots & p_{(k-1)(n} \end{array}\right] \quad (4.14)$$

onde $\mathbf{I}_k$ é a matriz identidade $k \times k$ e **P** é uma matriz $k \times (n-k)$ que determina os $n-k$ bits de paridade na palavra-código $\underline{c}_i$ de n bits, a partir dos k bits da mensagem $\underline{x}_i$.

Por exemplo:

Mensagem $\underline{x}_i$ de $k=3$ bits	Palavra-código $\underline{c}_i$ de $n=4$ bits associada a $\underline{x}_i$ por $\underline{c}_i = \boldsymbol{\theta}\{\underline{x}_i\}$.
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

A matriz geradora do Exemplo 4.2 (reproduzido ao lado), $\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$, está na forma sistemática e o código $\boldsymbol{\theta}(4,3)$ gerado é um código sistemático, i.e., cada palavra-código de n bits é formada pelos k bits da respectiva mensagem associada, acrescidos (por justaposição) de $n-k$ bits de paridade.

No contexto de comunicação digital, as palavras-código passam por um processo de modulação no trasmissor e são enviadas através de um canal com ruído/interferência.

Dois códigos que diferem somente na ordem (arranjo) de suas palavras-código, apresentam a mesma probabilidade de erro de decodificação no receptor, porque as distâncias de Hamming entre as palavras-código são as mesmas [Peterson]. Tais códigos são denominados **equivalentes**.

Especificamente, o código $\boldsymbol{\theta}_e(n,k)$ é equivalente ao código $\boldsymbol{\theta}(n,k)$ se a matriz geradora $\mathbf{G}_e$ de $\boldsymbol{\theta}_e(n,k)$ puder ser obtida através da permutação de **colunas** da matriz **G** geradora de $\boldsymbol{\theta}(n,k)$ ou através de operações elementares realizadas entre as **linhas** de **G**.

Uma operação elementar em $\mathbf{GF}(2)$ entre duas linhas de uma matriz consiste em permutar as linhas ou em substituir uma linha pela soma dela com outra linha.

Assim sempre podemos transformar uma matriz $\mathbf{G}$ qualquer para a forma sistemática $\mathbf{G}^*$ dada por (4.14) , mantendo a equivalência entre os respectivos códigos gerados.

Exemplo 4.3:

Dada a matriz geradora $\mathbf{G} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$, colocá-la na forma sistemática $\mathbf{G}^*$.

Verifique se $\mathbf{G}^*$ gera um código equivalente ao gerado por $\mathbf{G}$.

Solução:

Visto que a matriz geradora é uma matriz $\mathbf{G}_{3\times 4}$, então o código gerado será um código $\boldsymbol{\theta}(4,3)$.

$\mathbf{G}^*$ pode ser obtida pelo seguinte conjunto de operações elementares feito sobre as linhas de $\mathbf{G}$:

Operação Elementar	Matriz $\mathbf{G}$ alterada
$L_0 \leftrightarrow L_2$	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$L_0 \leftarrow (L_0 + L_1)$	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$L_0 \leftarrow (L_0 + L_2)$	$\mathbf{G}^* = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

O código gerado por **G** é

$\underline{x}_i$	$\underline{x}_i\mathbf{G} = \underline{c}_i$	$\underline{x}_i$	$\underline{x}_i\mathbf{G} = \underline{c}_i$
$[0\ 0\ 0]$	$[0\ 0\ 0]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[0\ 0\ 0\ 0]$	$[1\ 0\ 0]$	$[1\ 0\ 0]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[0\ 0\ 1\ 1]$
$[0\ 0\ 1]$	$[0\ 0\ 1]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[1\ 1\ 1\ 1]$	$[1\ 0\ 1]$	$[1\ 0\ 1]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[1\ 1\ 0\ 0]$
$[0\ 1\ 0]$	$[0\ 1\ 0]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[0\ 1\ 0\ 1]$	$[1\ 1\ 0]$	$[1\ 1\ 0]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[0\ 1\ 1\ 0]$
$[0\ 1\ 1]$	$[0\ 1\ 1]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[1\ 0\ 1\ 0]$	$[1\ 1\ 1]$	$[1\ 1\ 1]\begin{bmatrix}0&0&1&1\\0&1&0&1\\1&1&1&1\end{bmatrix}=[1\ 0\ 0\ 1]$

O código gerado por $\mathbf{G}^*$ é o mesmo código gerado no Exemplo 4.2.

Portanto os códigos gerados por $\mathbf{G}^*$ e $\mathbf{G}$ são equivalentes,

porque diferem apenas na ordem (arranjo) de suas palavras-código.

4.3.3 A Matriz de Paridade de um Código $\boldsymbol{\theta}(n,k)$

Seja um código $\boldsymbol{\theta}(n,k)$ com matriz geradora **G** dada na forma sistemática, isto é,

$$\mathbf{G} = \left[\mathbf{I}_k \vdots \mathbf{P}\right] \tag{4.19}$$

Conforme discutimos na Seção 4.3.2, a *i*-ésima palavra-código $\underline{c}_i = \left[c_{i0} \quad c_{i1} \quad \cdots \quad c_{i(n-1)}\right]$ relaciona-se com a respectiva mensagem $\underline{x}_i = \left[x_{i0} \quad x_{i1} \quad \cdots \quad x_{i(k-1)}\right]$ através de $\underline{c}_i = \underline{x}_i\mathbf{G}$.

Já que **G** encontra-se na forma sistemática, a palavra-código $\underline{c}_i$ pode ser decomposta em

$$\underline{c}_i = \left[\underline{x}_i \quad \underline{a}_i\right] \tag{4.20}$$

onde $\underline{a}_i = \underline{x}_i\mathbf{P}$ é um vetor-linha que contém os $n-k$ bits de paridade de $\underline{c}_i$.

Visto que $\underline{a}_i = \underline{x}_i\mathbf{P}$, e considerando que a soma em $\mathbf{GF}(2)$ é uma operação módulo 2 (ver Tab. 4.3), então

$$\underline{x}_i\mathbf{P} + \underline{a}_i = \underline{0} \tag{4.21}$$

que pode ser escrita matricialmente como

$$\left[\underline{x}_i \quad \underline{a}_i\right]\begin{bmatrix}\mathbf{P}\\ \mathbf{I}_{n-k}\end{bmatrix} = \underline{0} \tag{4.22}$$

Definindo $\mathbf{H}^T = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix}$ temos, de $\begin{bmatrix} \underline{x}_i & \underline{a}_i \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix} = \underline{0}$ (4.22) que

$$\underline{c}_i \mathbf{H}^T = \underline{0} \qquad (4.23)$$

sendo

$$\mathbf{H} = \left(\mathbf{H}^T\right)^T = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix}^T = \left[\mathbf{P}^T \;\vdots\; \left(\mathbf{I}_{n-k}\right)^T\right] = \left[\mathbf{P}^T \;\vdots\; \mathbf{I}_{n-k}\right] \qquad (4.24)$$

- Portanto, de $\underline{c}_i \mathbf{H}^T = \underline{0}$ (4.23), infere-se que cada palavra-código do código $\boldsymbol{\theta}(n,k)$ é ortogonal a cada linha da matriz **H** (se $\underline{u} \cdot \underline{v}^T = 0$ então os vetores $\underline{u}$ e $\underline{v}$ são ortogonais [Chen]).
- Em conseqüência, como as palavras-código do código $\boldsymbol{\theta}(n,k)$ são geradas por **G**, então

$$\mathbf{G}\mathbf{H}^T = \underline{0} \qquad (4.25)$$

⇒Observe que a matriz **H** pode ser usada no receptor digital para detectar e localizar em qual bit da palavra-código recebida ocorreu erro como conseqüência da degradação imposta pelo canal de transmissão.

⇒Sempre que a palavra-código $\underline{y}_i$ recebida no receptor digital resultar $\underline{y}_i \mathbf{H}^T \neq \underline{0}$ então $\underline{y}_i$ apresenta erros.

⇒Por este motivo, $\mathbf{H}_{(n-k)\times n}$ é denominada de **matriz de paridade**. Discutiremos esta propriedade na Seção 4.3.4.

Exemplo 4.4:

Determine a matriz de paridade **H** do código $\boldsymbol{\theta}(4,3)$ do Exemplo 4.3 e verifique se $\mathbf{G}\mathbf{H}^T = \underline{0}$ e se $\underline{c}_i \mathbf{H}^T = \underline{0}$.

Solução:

A matriz geradora de $\boldsymbol{\theta}(4,3)$ na forma sistemática é $\mathbf{G} = \left[\mathbf{I}_3 \vdots \mathbf{P}\right] = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$.

De (4.24) temos

$$\mathbf{H} = \left[\mathbf{P}^T \vdots \mathbf{I}_{n-k}\right] = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \qquad (4.26)$$

$$\mathbf{G}\mathbf{H}^T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad (4.27)$$

Verificando se $\underline{c}_i \mathbf{H}^T = \underline{0}$:

$\begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0 & 1 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$
$\begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}1 & 1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\end{bmatrix}$

4.3.4 Decodificação pela Mínima Distância

(Decodificacão MLD - *Maximum-Likelihood Decoding*)

Nesta seção estudaremos como os erros nas palavras-código são detectados e corrigidos no receptor digital.

- No receptor digital, os n bits provenientes do demodulador, correspondentes à i-ésima palavra-código $\underline{y}_i$ recebida são entregues ao decodificador do código $\boldsymbol{\theta}(n,k)$.
- O decodificador compara $\underline{y}_i$ com as $M = 2^k$ possíveis palavras-código $\underline{c}_j$ de $\boldsymbol{\theta}(n,k)$, $j = 0,1,\cdots,M-1$, e decide em favor daquela palavra-código (portanto, em favor da mensagem associada) que é mais próxima da palavra-código recebida em termos da **Distância de Hamming**.

Matematicamente esta operação pode ser expressa por

$$\boldsymbol{\theta}^{-1}\{\underline{y}_i\} = \arg\min_{\underline{c}_j} \left|\underline{y}_i - \underline{c}_j\right|_H \qquad (4.29)$$

onde $\underline{c}_j \in \mathbf{C}$,

$\mathbf{C} = \{\underline{c}_i\} = \{\underline{c}_0, \underline{c}_1, \cdots, \underline{c}_{M-1}\}$ e

$\left|\underline{y}_i - \underline{c}_j\right|_H$ denota a Distância de Hamming entre $\underline{y}_i$ e $\underline{c}_j$.

A decodificação com base na distância mínima é ótima no sentido de que resulta na mínima probabilidade de erro de decodificação se:

a) As palavras-código do conjunto $\mathbf{C} = \{\underline{c}_i\} = \{\underline{c}_0, \underline{c}_1, \cdots, \underline{c}_{M-1}\}$ apresentam distribuição de probabilidade uniforme (i.e., as palavras-código são equiprováveis).

b) O canal de transmissão não altera esta distribuição de probabilidade.

Um decodificador baseado no critério de distância mínima é denominado de **Decodificador de Máxima Verossimilhança** ou **Decodificador ML** (ML - *Maximum-Likelihood*).

Em geral, pelo menos uma das duas condições, (a) ou (b), não pode ser obedecida na prática.

Nesta situação, o decodificador ótimo passa a ser o **Decodificador MAP** (MAP - *maximum a posteriori*), muito embora este não seja encontrado com muita freqüência na implementação de sistemas de comunicações digitais.

Um decodificador MAP leva em conta a distribuição estatística das palavras-código e toma a decisão sobre qual palavra-código foi recebida com base em Estatística Bayesiana [Proakis].

Na prática, decodificadores MAP não são muito utilizados porque sua operação depende do conhecimento exato da distribuição de probabilidade das palavras-código e da variância de ruído no canal, informação quase sempre não disponível.

A maioria dos sistemas digitais utiliza decodificadores ML ao invés de decodificadores MAP principalmente devido à insignificante melhora obtida com o uso de um decodificador MAP às expensas de um considerável aumento da complexidade computacional do decodificador [Messerschmitt]. **Focalizaremos nossa atenção, portanto, em decodificadores ML.**

Embora a **decodificação ML** possa ser realizada através de $\boldsymbol{\theta}^{-1}\{\underline{y}_i\} = \arg\min_{\underline{c}_j} \left|\underline{y}_i - \underline{c}_j\right|_H$ (4.29), existe uma maneira mais eficiente de implementar um decodificador ML, aproveitando as propriedades da matriz de paridade $\mathbf{H}_{(n-k)\times n}$ de um código $\boldsymbol{\theta}(n,k)$, denominada de **Decodificação por Arranjo Padrão** (*Standard Array Decoding*).

A desvantagem de (4.29) é a necessidade de calcular $M = 2^k$ Distâncias de Hamming para decodificar a palavra-código recebida.

Veremos a seguir como reduzir este número de distâncias calculadas para 2^{n-k} utilizando o conceito de **Arranjo Padrão**, já que, na prática, usualmente $n-k < k$.

Arranjo Padrão

⇨ Seja $\underline{c}_i$ a palavra-código transmitida pelo transmissor digital através do canal de transmissão e seja $\underline{y}$ a palavra-código recebida resultante na saída do demodulador do receptor digital.

⇨ Devido à degradação do sinal no canal em conseqüência de ruído/interferência, a palavra-código $\underline{y}$ recebida pode conter erros, de modo que $\underline{y}$ pode ser expressa por

$$\underline{y} = \underline{c}_i + \underline{e} \qquad (4.30)$$

onde $\underline{e}$ é o vetor-linha de *n* bits que representa o **padrão de erro** (i.e., os bits errados em $\underline{y}$) resultante da degradação do sinal no canal.

⇨ Note que o peso do padrão de erro é a Distância de Hamming entre $\underline{y}$ e $\underline{c}_i$.

- Pós-multiplicando (4.30) por $\mathbf{H}^T$ obtemos

$$\underline{y}\mathbf{H}^T = (\underline{c}_i + \underline{e})\mathbf{H}^T = \underline{c}_i\mathbf{H}^T + \underline{e}\mathbf{H}^T = \underline{e}\mathbf{H}^T \qquad (4.31)$$

- Define-se o vetor $n-k$ dimensional $\underline{s}$, denominado **síndrome do padrão de erro**, ou simplesmente **síndrome**, como

$$\underline{s} = \underline{e}\mathbf{H}^T \qquad (4.32)$$

- É importante enfatizar que o conjunto de síndromes $\{\underline{s}\}$ é determinado pelo conjunto de padrões de erro $\{\underline{e}\}$ mas **não** pelo conjunto **C** de palavras-código transmitidas, como podemos inferir de (4.31).

Ainda, visto que

- $\underline{e}$ é um vetor de n bits (i.e., $\underline{e}$ é um vetor n dimensional em $\mathbf{GF}(2)$) então existem 2^n possíveis padrões de erro no conjunto $\{\underline{e}\}$ e
- $\underline{s}$ é um vetor de $n-k$ bits e, portanto, existem 2^{n-k} possíveis síndromes no conjunto $\{\underline{s}\}$.
- Em conseqüência, $\underline{s} = \underline{e}\mathbf{H}^T$ (4.32) mapeia diferentes padrões de erro $\underline{e}$ na mesma síndrome $\underline{S}$.

O Arranjo Padrão (AP) resulta em uma tabela, denominada **Tabela de Síndromes**, a qual é implementada em ROM (ROM - *Read Only Memory*) no receptor digital.

A Tabela de Síndromes é consultada pelo decodificador ML para identificação e correção de erro em cada palavra-código $\underline{y}$ recebida.

(Veremos como construir a Tabela de Síndromes no Exemplo 4.6. Por enquanto, focalizaremos nossa atenção na construção do AP, visto que a capacidade de detecção/correção de um código pode ser detalhadamente obtida a pertir do AP.)

O AP também é uma tabela que possui 2^{n-k} linhas, cada uma delas associada a uma das 2^{n-k} possíveis síndromes.

O número de colunas do AP é 2^k, correspondendo ao número de palavras-código do código $\boldsymbol{\theta}(n,k)$. Quando implementado manualmente, a linha superior do AP recebe a designação de L0 e a coluna mais à esquerda recebe a designação C0.

A Tabela 4.5 mostra a forma geral de um AP, o qual, portanto, é formado de $2^{n-k} \times 2^k = 2^n$ células.

Tabela 4.5: Forma geral de um Arranjo Padrão

	C0	C1	C2	…	C($2^k - 1$)
L0	$\underline{e}_0 = \underline{c}_0 = \underline{0}$	$\underline{c}_1$	$\underline{c}_2$	…	$\underline{c}_{(2^k-1)}$
L1	$\underline{e}_1$	$\underline{c}_1 + \underline{e}_1$	$\underline{c}_2 + \underline{e}_1$	…	$\underline{c}_{(2^k-1)} + \underline{e}_1$
L2	$\underline{e}_2$	$\underline{c}_1 + \underline{e}_2$	$\underline{c}_2 + \underline{e}_2$	…	$\underline{c}_{(2^k-1)} + \underline{e}_2$
⋮	⋮	⋮	⋮		⋮
L($2^{n-k} - 1$)	$\underline{e}_{(2^{n-k}-1)}$	$\underline{c}_1 + \underline{e}_{(2^{n-k}-1)}$	$\underline{c}_2 + \underline{e}_{(2^{n-k}-1)}$	…	$\underline{c}_{(2^k-1)} + \underline{e}_{(2^{n-k}-1)}$

Tabela 4.5: Forma geral de um Arranjo Padrão					
	C0	C1	C2	…	C(2^k-1)
L0	$\underline{e}_0 = \underline{c}_0 = \underline{0}$	$\underline{c}_1$	$\underline{c}_2$	…	$\underline{c}_{(2^k-1)}$
L1	$\underline{e}_1$	$\underline{c}_1 + \underline{e}_1$	$\underline{c}_2 + \underline{e}_1$	…	$\underline{c}_{(2^k-1)} + \underline{e}_1$
L2	$\underline{e}_2$	$\underline{c}_1 + \underline{e}_2$	$\underline{c}_2 + \underline{e}_2$	…	$\underline{c}_{(2^k-1)} + \underline{e}_2$
⋮	⋮	⋮	⋮		⋮
L($2^{n-k}-1$)	$\underline{e}_{(2^{n-k}-1)}$	$\underline{c}_1 + \underline{e}_{(2^{n-k}-1)}$	$\underline{c}_2 + \underline{e}_{(2^{n-k}-1)}$	…	$\underline{c}_{(2^k-1)} + \underline{e}_{(2^{n-k}-1)}$

Na linha L0 do AP são listadas, da esquerda para a direita, as 2^k palavras-código de $\boldsymbol{\theta}(n,k)$, cada uma delas representada por um vetor *n* dimensional em $\mathbf{GF}(2)$.

A palavra-código $\underline{c}_0$ pertencente à célula identificada pela intersecção da coluna C0 com a linha L0 (célula $\mathrm{L0}\times\mathrm{C0}$) obrigatoriamente deve ser aquela representada pelo vetor $\underline{0}$.

Na coluna C0, abaixo da palavra-código $\underline{0}$, são listados, de alto a baixo, os $2^{n-k}-1$ padrões de erro relativos à palavra-código $\underline{c}_0 = \underline{0}$.

Primeiramente são listados todos os *n* padrões de erro de peso 1, isto é, todos os padrões de erro que resultam de uma Distância de Hamming unitária entre a palavra-código $\underline{y}$ recebida e $\underline{c}_0 = \underline{0}$.

Se $2^{n-k} > n$, então lista-se a seguir em C0 todos os possíveis padrões de erro de peso 2.

Em seguida lista-se em C0 todos os possíveis padrões de erro de peso 3, e assim sucessivamente até que todas as 2^{n-k} células de C0 estejam preenchidas.

Neste contexto, $\underline{e}_0 = \underline{c}_0 = \underline{0}$ representa o padrão de erro de peso 0, isto é, representa a não-ocorrência de erro.

Nota: Visto que cada linha do AP necessita corresponder a uma única síndrome dentre as 2^{n-k} possíveis síndromes, devemos ter o cuidado de, na construção de C0, assegurar que distintos padrões de erro de peso maior que 1 em C0 correspondam a síndromes que são distintas entre si e que são simultaneamente distintas daquelas que correspondem a padrões de erro de peso 1.

Então, dando prosseguimento à construção do AP, adicionamos o padrão de erro contido na *i*-ésima célula de C0 à palavra-código na célula $\mathrm{L0}\times\mathrm{C1}$ e colocamos o resultado na *i*-ésima célula em C1. Em seguida, adicionamos o padrão de erro contido na *i*-ésima célula de C0 à palavra-código na célula $\mathrm{L0}\times\mathrm{C2}$ e colocamos o resultado na *i*-ésima célula em C2, e assim sucessivamente até completar a última coluna $\mathrm{C}(2^k-1)$, mais à direita do AP, sendo $i = 0,1,2,\cdots,2^{n=k}-1$.

Observe que a *i*-ésima linha L*i* do AP assim construído (incluindo L0) representa o conjunto das 2^k possíveis palavras-código $\underline{y}_j = \underline{c}_j + \underline{e}_i$, $j = 0,1,\cdots,2^k-1$, que serão recebidas caso a degradação do canal gere o padrão de erro $\underline{e}_i$ contido na célula $\mathrm{L}i\times\mathrm{C0}$.

Cada linha L*i* do AP é denominada de ***coset*** e a célula $\mathrm{L}i\times\mathrm{C0}$ é denominada **líder do *coset***. Portanto, um *coset* é o conjunto de todas as palavras-código possíveis de serem recebidas quando o canal impõe o padrão de erro definido pelo líder do *coset*.

Exemplo 4.6:

Seja o codificador de canal no transmissor de um sistema de comunicação digital que utiliza o código de bloco gerado por $\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$.

a) Determine um possível AP para este código e a Tabela de Síndromes associada, visando o projeto do decodificador no receptor.

b) Suponha que o transmissor digital envie a palavra-código $\underline{c} = [1 \quad 0 \quad 1 \quad 0 \quad 1]$ através do canal. O canal degrada o sinal de forma que o demodulador no receptor envia para o decodificador a palavra-código $\underline{y} = [1 \quad 1 \quad 1 \quad 0 \quad 1]$ (erro no bit b_3). Verifique a capacidade do decodificador em detectar e corrigir este erro.

c) Suponha que o ruído/interferência no canal seja alto de forma que o demodulador no receptor envia para o decodificador a palavra-código $\underline{y} = [1 \quad 1 \quad 1 \quad 1 \quad 1]$ (erro no bit b_0 e no b_3). Verifique a capacidade do decodificador em detectar e corrigir este erro duplo.

Solução:

a) Visto que $\mathbf{G}_{k \times n} = \mathbf{G}_{2 \times 5}$, o código em questão é $\boldsymbol{\theta}(5,2)$.

As $2^k = 2^2 = 4$ palavras-código de $\boldsymbol{\theta}(5,2)$ gerado por $\mathbf{G}$ são obtidas de (4.10):

$$\underline{c}_0 = [0 \quad 0]\mathbf{G} = [0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\underline{c}_1 = [0 \quad 1]\mathbf{G} = [0 \quad 1 \quad 0 \quad 1 \quad 1]$$

$$\underline{c}_2 = [1 \quad 0]\mathbf{G} = [1 \quad 0 \quad 1 \quad 0 \quad 1]$$

$$\underline{c}_3 = [1 \quad 1]\mathbf{G} = [1 \quad 1 \quad 1 \quad 1 \quad 0]$$

A matriz geradora não necessita ser transformada por permutação de colunas ou por operações elementares em linhas visto que já encontra-se na forma sistemática, isto é, $\mathbf{G} = \left[\begin{array}{ccc|cc} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right] = [\mathbf{I}_2 \mid \mathbf{P}]$.

Daí, de (4.24) temos que $\mathbf{H}_{(n-k)\times n} = [\mathbf{P}^T \mid \mathbf{I}_{n-k}] = \left[\begin{array}{cc|ccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{array}\right]$.

Para determinar os padrões de erro da coluna C0 do AP precisamos verificar quais as síndromes resultantes dos $n = 5$ padrões de erro de peso 1 para que não ocorra igualdade com as síndromes resultantes dos padrões de erro de peso maior que 1. Os padrões de erro de peso 1 são: $[0\ 0\ 0\ 0\ 1], [0\ 0\ 0\ 1\ 0], [0\ 0\ 1\ 0\ 0], [0\ 1\ 0\ 0\ 0]$ e $[1\ 0\ 0\ 0\ 0]$.

Verificando as síndromes resultantes dos padrões de erro de peso 1:

$\underline{e}_i$	$\underline{e}_i \mathbf{H}^T = \underline{s}_i$
$[0\ 0\ 0\ 0\ 1]$	$[0\ 0\ 0\ 0\ 1]\mathbf{H}^T = [0\ 0\ 1]$
$[0\ 0\ 0\ 1\ 0]$	$[0\ 0\ 0\ 1\ 0]\mathbf{H}^T = [0\ 1\ 0]$
$[0\ 0\ 1\ 0\ 0]$	$[0\ 0\ 1\ 0\ 0]\mathbf{H}^T = [1\ 0\ 0]$
$[0\ 1\ 0\ 0\ 0]$	$[0\ 1\ 0\ 0\ 0]\mathbf{H}^T = [0\ 1\ 1]$
$[1\ 0\ 0\ 0\ 0]$	$[1\ 0\ 0\ 0\ 0]\mathbf{H}^T = [1\ 0\ 1]$

Obviamente a síndrome resultante do padrão de erro de peso 0 (inexistência de erro) é $[0\ 0\ 0\ 0\ 0]\mathbf{H}^T = [0\ 0\ 0]$.

O AP a ser construído possui $2^{n-k} = 2^{5-2} = 8$ linhas (correspondentes às 2^{n-k} síndromes). Já determinamos $n+1=6$ síndromes. Ainda faltam determinar $2^{n-k} - (n+1) = 8-(5+1) = 2$ síndromes. Estas 2 síndromes faltantes devem **obrigatoriamente** ser distintas entre si e distintas das $n+1=6$ síndromes já determinadas. Tendo esta condição em mente, verifica-se na tabela acima que elas são as síndromes $[1 \;\; 1 \;\; 0]$ e $[1 \;\; 1 \;\; 1]$.

Os padrões de erro que resultam nestas 2 síndromes (que estamos buscando determinar para formar a coluna C0 do AP) devem ser padrões de erro de peso 2, visto que já esgotamos os possíveis padrões de erro de peso 0 e de peso 1.

Se expressarmos o padrão de erro por $\underline{e} = [b_4 \;\; b_3 \;\; b_2 \;\; b_1 \;\; b_0]$, onde b_i representa a ordem do bit, e considerando que $\underline{s} = \underline{e}\mathbf{H}^T$ (Equação (4.32)), temos que para a síndrome $[1 \;\; 1 \;\; 0]$:

$$[1 \;\; 1 \;\; 0] = [b_4 \;\; b_3 \;\; b_2 \;\; b_1 \;\; b_0] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

o que resulta no seguinte sistema de equações em $\mathbf{GF}(2)$:

$$b_4 + b_2 = 1 \rightarrow b_4 = b_2 + 1 \rightarrow b_4 = \overline{b_2}$$

$$b_3 + b_1 = 1 \rightarrow b_3 = b_1 + 1 \rightarrow b_3 = \overline{b_1}$$

$$b_4 + b_3 + b_0 = 0 \rightarrow b_2 + 1 + b_1 + 1 + b_0 = 0 \rightarrow b_2 + b_1 + b_0 = 0$$

onde $\overline{b}$ representa a negação do valor lógico do bit *b*. Um possível padrão de erro de peso 2 que obedece às equações acima é $\underline{e} = [1 \;\; 1 \;\; 0 \;\; 0 \;\; 0]$. Portanto este será o padrão de erro que associaremos à síndrome $[1 \;\; 1 \;\; 0]$.

Para a síndrome $[1 \quad 1 \quad 0]$ temos que:

$$[1 \quad 1 \quad 1] = [b_4 \quad b_3 \quad b_2 \quad b_1 \quad b_0] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

o que resulta no seguinte sistema de equações em $\mathbf{GF}(2)$:

$$b_4 + b_2 = 1 \rightarrow b_4 = b_2 + 1 \rightarrow b_4 = \overline{b_2}$$

$$b_3 + b_1 = 1 \rightarrow b_3 = b_1 + 1 \rightarrow b_3 = \overline{b_1}$$

$$b_4 + b_3 + b_0 = 1 \rightarrow b_2 + 1 + b_1 + 1 + b_0 = 1 \rightarrow b_2 + b_1 + b_0 = 1$$

Um possível padrão de erro de peso 2, distinto do anterior, que obedece às equações acima é $\underline{e} = [1 \quad 0 \quad 0 \quad 1 \quad 0]$. Portanto este será o padrão de erro que associaremos à síndrome $[1 \quad 1 \quad 1]$.

De posse destes resultados, o AP é construído como:

Arranjo Padrão:				
	C0	C1	C2	C3
L0	[0 0 0 0 0]	[0 1 0 1 1]	[1 0 1 0 1]	[1 1 1 1 0]
L1	[0 0 0 0 1]	[0 1 0 1 0]	[1 0 1 0 0]	[1 1 1 1 1]
L2	[0 0 0 1 0]	[0 1 0 0 1]	[1 0 1 1 1]	[1 1 1 0 0]
L3	[0 0 1 0 0]	[0 1 1 1 1]	[1 0 0 0 1]	[1 1 0 1 0]
L4	[0 1 0 0 0]	[0 0 0 1 1]	[1 1 1 0 1]	[1 0 1 1 0]
L5	[1 0 0 0 0]	[1 1 0 1 1]	[0 0 1 0 1]	[0 1 1 1 0]
L6	[1 1 0 0 0]	[1 0 0 1 1]	[0 1 1 0 1]	[0 0 1 1 0]
L7	[1 0 0 1 0]	[1 1 0 0 1]	[0 0 1 1 1]	[0 1 1 0 0]

E a Tabela de Síndromes para implentação do decodificador é:

Tabela de Síndromes (implementada em ROM):	
Síndrome $\underline{s}_i$	Padrão de Erro $\underline{e}_i$
$[0 \quad 0 \quad 0]$	$[0 \quad 0 \quad 0 \quad 0 \quad 0]$
$[0 \quad 0 \quad 1]$	$[0 \quad 0 \quad 0 \quad 0 \quad 1]$
$[0 \quad 1 \quad 0]$	$[0 \quad 0 \quad 0 \quad 1 \quad 0]$
$[0 \quad 1 \quad 1]$	$[0 \quad 1 \quad 0 \quad 0 \quad 0]$
$[1 \quad 0 \quad 0]$	$[0 \quad 0 \quad 1 \quad 0 \quad 0]$
$[1 \quad 0 \quad 1]$	$[1 \quad 0 \quad 0 \quad 0 \quad 0]$
$[1 \quad 1 \quad 0]$	$[1 \quad 1 \quad 0 \quad 0 \quad 0]$
$[1 \quad 1 \quad 1]$	$[1 \quad 0 \quad 0 \quad 1 \quad 0]$

b) De (4.31) temos que $\underline{y}\mathbf{H}^T = \underline{e}\mathbf{H}^T = \underline{s}$. Dado $\underline{y} = [1 \quad 1 \quad 1 \quad 0 \quad 1]$, então $\underline{s} = \underline{y}\mathbf{H}^T = [0 \quad 1 \quad 1]$. Consultando a Tabela de Síndromes verifica-se que o padrão de erro correspondente é $\underline{e} = [0 \quad 1 \quad 0 \quad 0 \quad 0]$. De (4.30) , $\underline{c}_i = \underline{y} + \underline{e} = [1 \quad 0 \quad 1 \quad 0 \quad 1]$. Portanto o decodificador detectou e corrigiu o erro simples.

c) De (4.31) temos que $\underline{y}\mathbf{H}^T = \underline{e}\mathbf{H}^T = \underline{s}$. Dado $\underline{y} = [1 \quad 1 \quad 1 \quad 1 \quad 1]$, então $\underline{s} = \underline{y}\mathbf{H}^T = [0 \quad 0 \quad 1]$. Consultando a Tabela de Síndromes verifica-se que o padrão de erro correspondente é $\underline{e} = [0 \quad 0 \quad 0 \quad 0 \quad 1]$. De (4.30) , $\underline{c}_i = \underline{y} + \underline{e} = [1 \quad 1 \quad 1 \quad 1 \quad 0]$. Portanto o decodificador detectou o erro mas **não** corrigiu o erro duplo.

A impossibilidade deste código corrigir todos os padrões de erro com peso maior que 1 pode ser também verificada bastando consultar a coluna C0 do AP. Por inspeção da coluna C0 conclue-se que este código corrige todos os 5 padrões de erro de peso 1 possíveis e somente 2 padrões de erro de peso 2, quais sejam, $\underline{e} = [1 \quad 1 \quad 0 \quad 0 \quad 0]$ e $\underline{e} = [1 \quad 0 \quad 0 \quad 1 \quad 0]$.

Em geral o projetista do código escolhe os padrões de erro de peso *w* que corrigem *w* erros de modo incompleto com base em alguma peculiaridade do sistema digital. Por exemplo, no Exemplo 4.6 o número total de padrões de erro de peso 2 é dado pela combinação de $n = 5$ bits tomados $m = 2$ a *m*, i.e. $\text{Comb}(n,m) = \text{Comb}(5,2) = 10$, onde $\text{Comb}(n,m) = n!/\,[m!(n-m)!]$. No entanto, na construção do AP foi possível utilizar apenas 2 deles: $\underline{e} = [1 \quad 1 \quad 0 \quad 0 \quad 0]$ e $\underline{e} = [1 \quad 0 \quad 0 \quad 1 \quad 0]$. A razão da escolha destes dois padrões poderia ser, por exemplo, o fato de que o bit b_4 é um bit crucial à supervisão e controle do sistema (supondo que o código seja sistemático) e que, em menor grau, o b_3 também o seja.

4.3.5 Códigos Estendidos

Qualquer código $\boldsymbol{\theta}(n,k)$ pode ser **estendido**, isto é, a partir da matriz de paridade $\mathbf{H}$ de $\boldsymbol{\theta}(n,k)$ a matriz estendida $\mathbf{H}_E$ é obtida de $\mathbf{H}$, conforme segue:

$$\mathbf{H}_E = \left[\begin{array}{ccc:c} & \mathbf{H} & & 0 \\ & & & \vdots \\ & & & 0 \\ \hdashline 1 & 1 & \cdots & 1 \end{array}\right] \qquad (4.33)$$

Demonstra-se que a distância mínima de um código estendido é igual à distância mínima do código não-estendido acrescida de 1, isto é, $d_{\min}E = d_{\min} + 1$ [Clark].

4.3.6 Principais Códigos de Bloco Binários

Há uma extensa coleção de códigos de bloco binários (e não binários). Entre eles citamos:

- **Códigos de Hadamard**

$\boldsymbol{\theta}(n,k) = \boldsymbol{\theta}(2^m, m+1)$, caracterizados por $d_{\min} = m+1$, onde $m \geq 1$ é um número inteiro.

Em geral, os Códigos de Hadamard apresentam baixa razão de codificação $R_c = k/n = \tau_s/\tau_x = (m+1)/2^m$, onde τ_s representa a largura (duração no tempo) dos bits em uma palavra-código e τ_x representa a largura dos bits na respectiva mensagem.

Portanto, como τ_s/τ_x é pequeno, o uso de um Código de Hadamard implica em um considerável **aumento** na banda-passante do sistema, e, por isso, não é muito utilizado.

- **Código de Golay**

$\boldsymbol{\theta}(23,12)$, caracterizado por $d\,\min = 7$, o que significa:

- uma capacidade de correção de até $t = \left\lfloor \frac{d\,\min - 1}{2} \right\rfloor = \left\lfloor \frac{7-1}{2} \right\rfloor = 3$ erros simultâneos e

- uma capacidade de detecção de até $d = d_{\min} - 1 = 7 - 1 = 6$ erros simultâneos.

Este código é peculiar porque ele é o único código conhecido de 23 bits capaz de corrigir até 3 erros simultâneos [Taub].

- **Códigos de Hamming**

$\boldsymbol{\theta}\left(2^m-1, 2^m-1-m\right)$, bastante populares por serem caracterizados pela extrema facilidade de construção aliada a uma distância mínima $d\min = 3$ [Peterson][Proakis] (detecta até 2 erros simultâneos e corrige até 1 erro), sendo $m = n-k$ um inteiro positivo. Por exemplo, se $m = 3$, obtemos um Código de Hamming $\boldsymbol{\theta}(7,4)$.

Em geral, a construção de um código de bloco $\boldsymbol{\theta}(n,k)$ consiste em definirmos a sua matriz de paridade $\mathbf{H}_{(n-k)\times n}$ e, a seguir, a partir de **H** (e de $\mathbf{G} = \left[\mathbf{I}_k \;\vdots\; \mathbf{P}\right] = \left[\begin{array}{ccccc|cccc} 1 & 0 & 0 & \cdots & 0 & p_{00} & p_{01} & \cdots & p_{0(n-k-1)} \\ 0 & 1 & 0 & \cdots & 0 & p_{10} & p_{11} & \cdots & p_{1(n-k-1)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{(k-1)0} & p_{(k-1)1} & \cdots & p_{(k-1)(n-k-1)} \end{array}\right]$), obtermos a sua matriz geradora $\mathbf{G}_{k\times n}$.

A matriz **H** de um Código de Hamming caracteriza-se pelas suas $n = 2^m - 1$ colunas serem formadas por todos os vetores distintos *m* dimensionais em $\mathbf{GF}(2)$, exceto o vetor $\underline{0}$.

Por exemplo, um código $\boldsymbol{\theta}(7,4)$ é um Código de Hamming com $m = 3$ em que a matriz **H** é formada pelos $n = 7$ vetores colunas $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$, $\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$, $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & 1 & 1\end{bmatrix}^T$, $\begin{bmatrix}1 & 1 & 0\end{bmatrix}^T$, $\begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$ e $\begin{bmatrix}1 & 0 & 1\end{bmatrix}^T$.

4.4 Códigos Reed-Solomon

Os Códigos Reed-Solomon constituem uma sub-classe de uma ampla classe de códigos cíclicos denominada de Códigos BCH (Bose – Chaudhuri – Hocquenghem).

Os Códigos Reed-Solomon (RS) encontram-se entre os códigos mais poderosos no que diz respeito à capacidade de correção de erro, sendo largamente utilizados em muitos sistemas digitais como:

- Dispositivos de armazenamento (Fita Magnética,CDs, DVD, códigos de barra, etc.).
- Comunicações Móveis e *wireless* (Telefonia celular, *links* de microondas, etc.)
- Comunicações via Satélite.
- Televisão Digital

Vimos anteriormente que um código de bloco binário $\boldsymbol{\theta}(n,k)$

codifica mensagens de *k* **bits** em palavras-código de *n* **bits**,

podendo corrigir até $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$ **bits** errados.

Um Código Reed-Solomon $\boldsymbol{\theta}(n,k)$, representado por $\mathbf{RS}(n,k)$,

codifica mensagens de *k* **símbolos** em palavras-código de *n* **símbolos**, sendo capaz de corrigir até $t = \left\lfloor \frac{n-k}{2} \right\rfloor$ **símbolos** errados.

Cada **símbolo** em uma palavra-código (ou em uma mensagem) de um código $\mathbf{RS}(n,k)$ é um bloco de *m* bits.

Daí, portanto, o poder de correção de erro de um código $\mathbf{RS}(n,k)$: Mesmo que **todos** os *m* bits de cada um dos *t* símbolos recebidos estejam errados, o código $\mathbf{RS}(n,k)$ efetua a correção não importando a localização dos símbolos na palavra-código.

Ainda, não importando o número e a posição dos bits errados em cada símbolo, o código $\mathbf{RS}(n,k)$ corrigirá até *t* símbolos e, caso o número de símbolos errados ultrapassar *t*, o código $\mathbf{RS}(n,k)$ detectará esta situação.

No contexto do codificador de canal de um sistema de comunicações digitais esta característica é extremamente vantajosa porque permite a correção de um surto de $m \times t$ bits sequenciais recebidos em erro (*error burst correction*).

Se o número de erros ultrapassar *t*, então o código $\mathbf{RS}(n,k)$ avisa o sistema de que não foi capaz de corrigir todos os erros.

É de especial interesse o caso em que $m = 8$, quando cada símbolo representa 1 *byte*.

Um *byte* representa um bloco de 8 bits, que é o menor bloco de informação usualmente encontrado em sistemas microprocessados.

Por exemplo, consideremos um código $\mathbf{RS}(20,16)$ com $m = 8$. Suponhamos que queiramos codificar a mensagem de $k = 16$ bytes:

255	100	012	098	120	003	233	111	077	163	000	001	088	200	101	007

O código $\mathbf{RS}(20,16)$ adiciona $n - k = 4$ bytes de paridade e codifica a mensagem acima na palavra-código em forma sistemática abaixo:

255	100	012	098	120	003	233	111	077	163	000	001	088	200	101	007	208	107	221	076

Observe que nenhum símbolo é maior do que 255, valor máximo decimal para 1 byte.

Observe também que as operações entre polinômios são todas executadas em $\mathbf{GF}(2^m) = \mathbf{GF}(2^8) = \mathbf{GF}(256)$.

Foge ao escopo deste texto o estudo da álgebra de polinômios em $\mathbf{GF}(2^m)$, e, portanto, não nos aprofundaremos na teoria dos Códigos Reed-Solomon.

4.5 Códigos Convolucionais – Decodificador de Viterbi

Um código convolucional é gerado pela combinação linear em $\mathbf{GF}(2)$ das saídas de um *shift-register* de *K* estágios.

A seqüência de bits a ser codificada é aplicada na entrada do *shift-register*, e este executa a convolução em $\mathbf{GF}(2)$ entre a seqüência de entrada e a resposta ao impulso da **máquina de estado** (*state machine*) representada pelo *shift-register*.

A saída da máquina de estado constitui, portanto, a seqüência codificada.

O número de estados da máquina de estado de um codificador convolucional é 2^K, sendo *K* o número de estágios do *shift-register*.

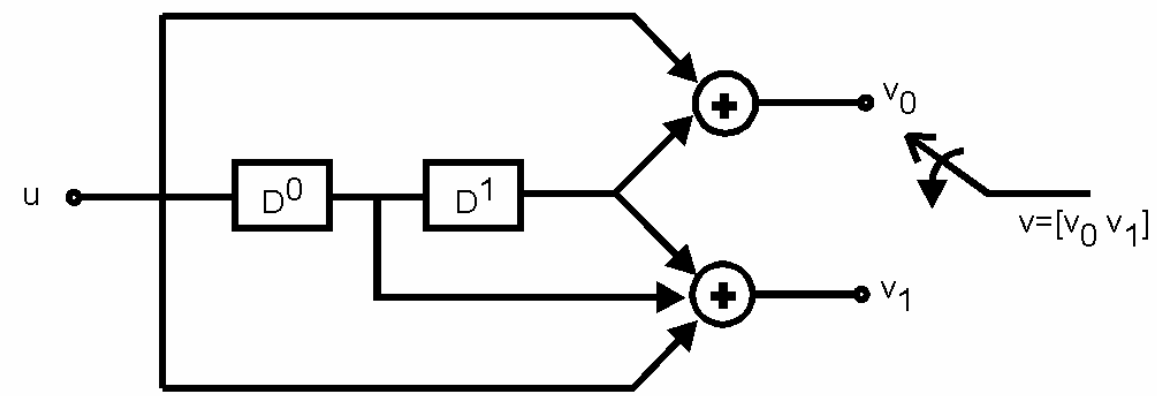
No contexto de códigos convolucionais *K* recebe o nome de *constraint length* [Taub].

A razão entre o número de entradas da máquina de estado e o número de saídas da mesma define a razão de codificação R_c do codificador.

Como uma máquina de estado construída a partir de um *shift-register* apresenta um conjunto finito de transições permitidas entre estados, quando a seqüência a ser codificada é a ela submetida, implicitamente ficarão restringidas as transições da seqüência codificada em sua saída.

Se o receptor conhecer a tabela de transições permitidas, então os erros gerados por degradação do sinal no canal de comunicações poderão ser identificados e corrigidos.

A Figura 4.2, abaixo, mostra um codificador convolucional com $K = 2$ ($2^K = 4$ estados) e $R_c = 1/2$.



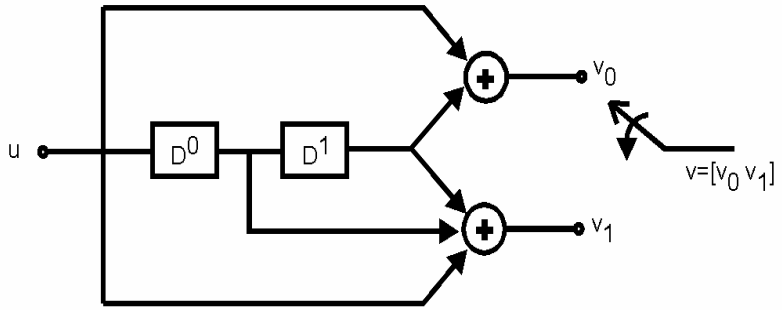
A seqüência de bits a ser codificada é representada por u e a saída do codificador é a seqüência de bits v.

Visto que $R_c = 1/2$, para cada bit de u são gerados dois bits em v.

O estágio D^0 transfere o valor lógico em sua entrada para a sua saída **imediatamente após** a ocorrência da borda de descida do pulso de *clock* (não representado na figura).

De forma idêntica opera o estágio D^1.

Representando a saída do estágio D^0 por D^0 e representando a saída do estágio D^1 por D^1, então o par de bits D^0D^1 identifica um dos estados da máquina de estado.



A Figura 4.3 mostra o diagrama de transição de estados do codificador convolucional da Figura acima.

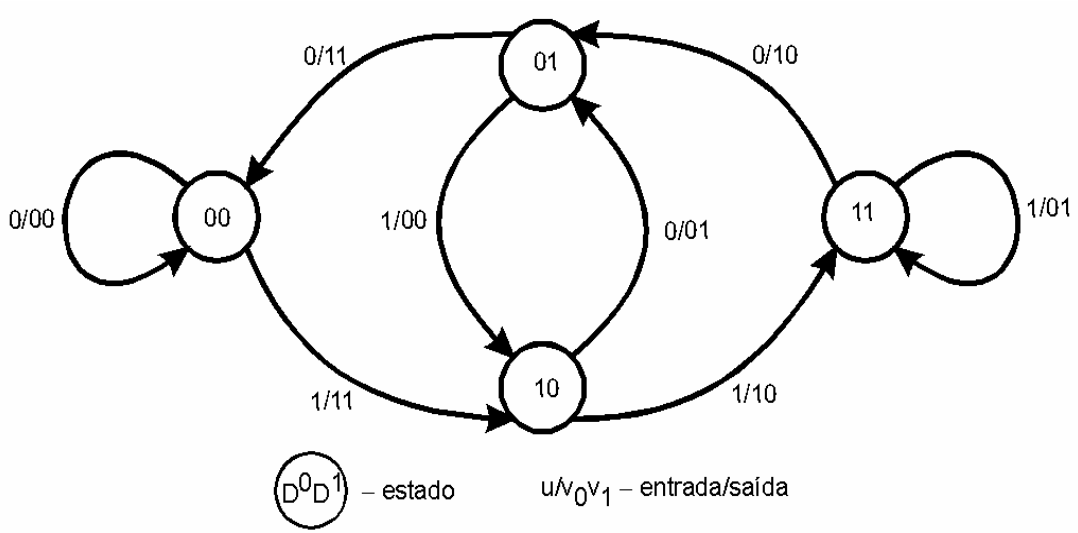
Cada círculo representa um estado D^0D^1 dentre os $2^K = 4$ possíveis estados.

O diagrama é construído a partir dos estados individuais considerando as **transições permitidas** a partir de cada estado como conseqüência do valor lógico de u.

Por exemplo, suponhamos que a máquina de estado encontre-se no estado 10 (i.e., $D^0 = 1$ e $D^1 = 0$ na Figura 4.2).

Se $u = 1$ a saída resultante é $v = 10$ e, após a borda de descida do *clock*, a máquina vai para o estado 11.

Por outro lado, se $u = 0$ a saída resultante é $v = 01$ e, após a borda de descida do *clock*, a máquina vai para o estado 01.



Dada uma seqüência u a ser codificada, a saída v no codificador de um transmissor digital é enviada ao receptor através do canal de transmissão, sendo recebida como uma seqüência r .

Se nenhuma degradação de sinal ocorreu no canal de transmissão, r= v.

A Tabela 4.8 mostra uma possível seqüência u e a resultante seqüência v para o codificador da Figura 4.2.

É mostrada também a trajetória do estado D^0D^1 a medida que u é codificada, partindo inicialmente do estado 00.

Assumindo que v seja enviado através de um canal de transmissão com ruído/interferência, a Tabela 4.8 mostra uma possível seqüência r recebida com 2 erros.

Tabela 4.8: Exemplo de codificação para o codificador da Figura 4.2

u =	0	1	1	0	0
D^0D^1 =	00	10	11	01	00
v =	00	11	10	10	11
r =	0***1***	11	1***1***	10	11

No receptor digital, o decodificador utiliza um algoritmo de decodificação baseado no princípio de mínima distância (MLSE – *maximum likelihood sequence detector*) denominado Algoritmo de Viterbi.

Vamos decodificar a seqüência r da Tabela 4.8 através do Algoritmo de Viterbi para testar a capacidade de correção de erros do mesmo.

A Figura 4.4 mostra o diagrama de treliça utilizado pelo Decodificador de Viterbi adequado ao codificador convolucional da Figura 4.2.

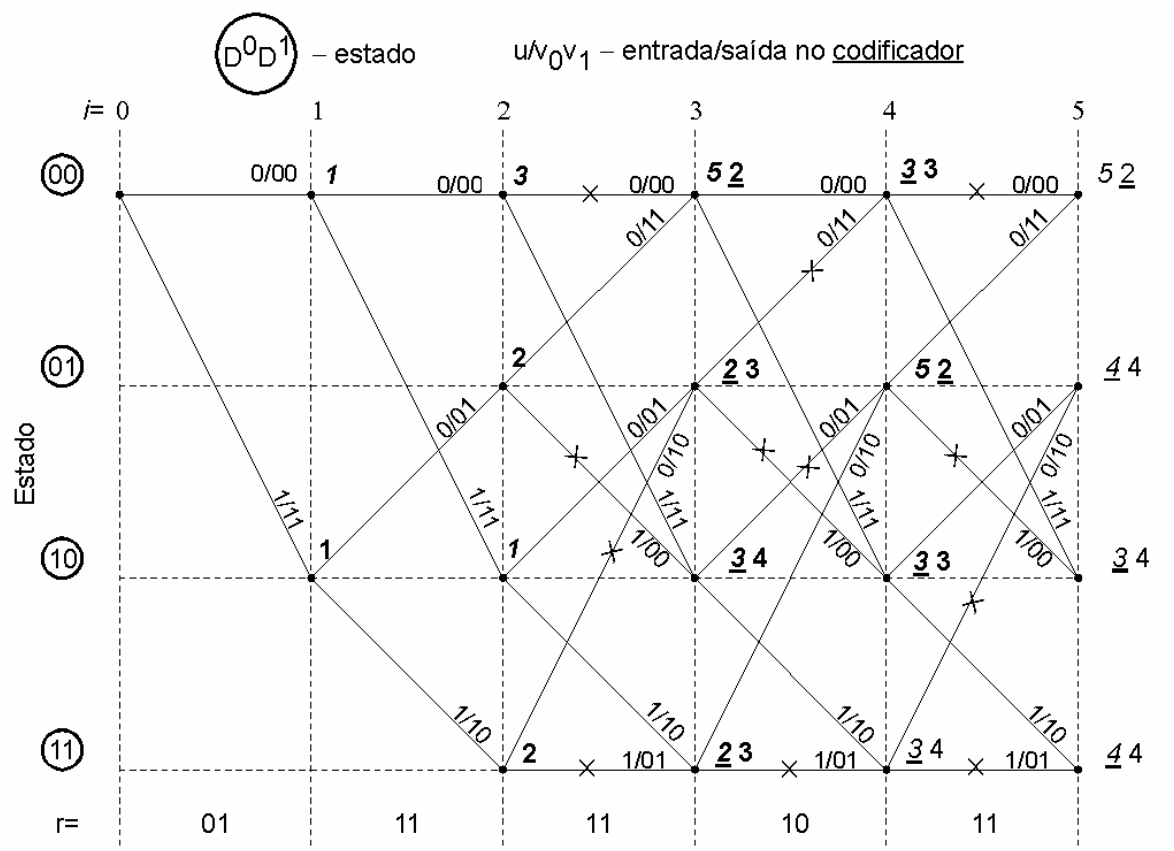
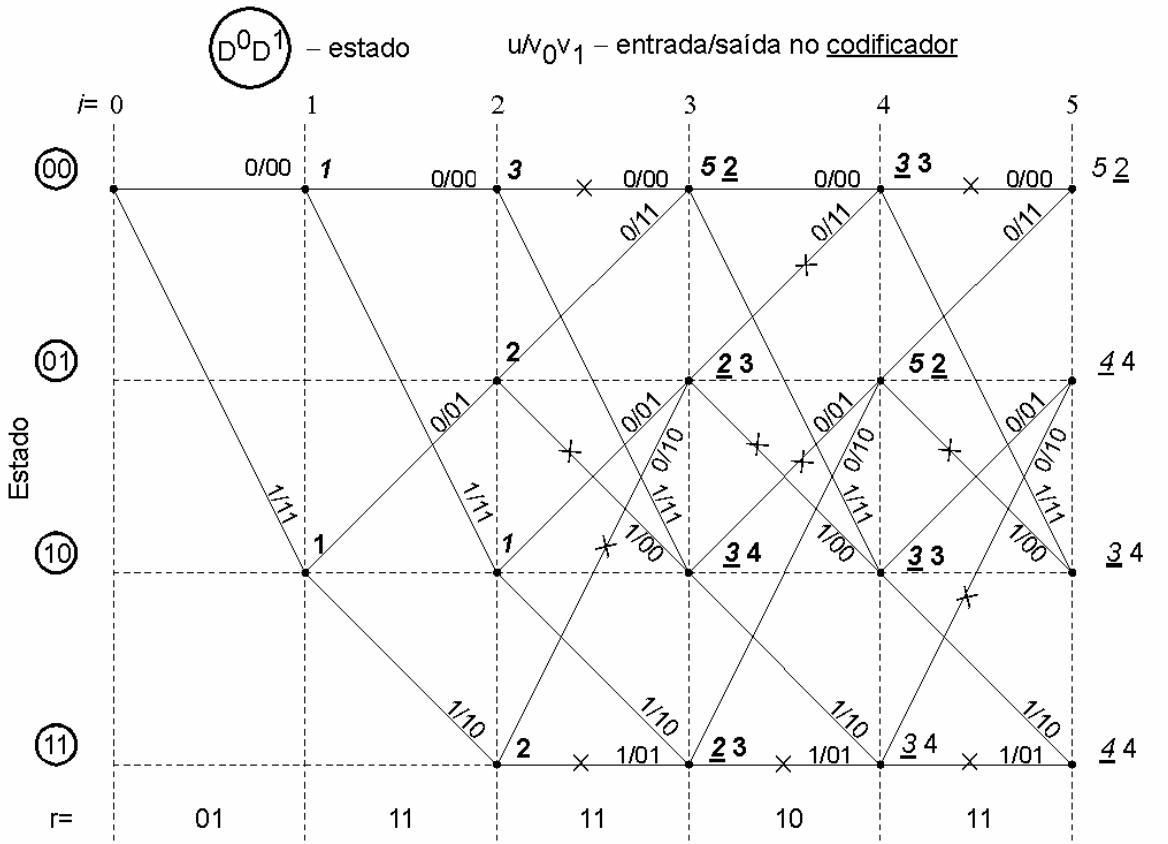


Diagrama de Treliça do Decodificador de Viterbi para o codificador convolucional da Figura 4.2.

O diagrama de treliça mostra todas as trajetórias (caminhos) das transições de estado da máquina de estado do codificador a cada instante *i* de codificação, a partir do estado 00.

Cada ramo da treliça começa e termina em um estado, representando, assim, uma transição permitida.

Cada ramo é identificado por u/v_0v_1, isto é, a saída v do codificador quando, ao aplicarmos u em sua entrada, a máquina de estado executa a transição representada pelo ramo em questão.



A técnica de decodificação consiste em acumular em cada nó da treliça as Distâncias de Hamming entre a saída v do codificador e a seqüência r recebida a cada instante *i*.

Se mais de um caminho chega a um nó "mata-se" aqueles de maior **métrica** (maior distância acumulada) – caminhos marcados com × na Figura 4.4 – ficando apenas aquele de menor métrica, denominado de **caminho sobrevivente**.

A métrica acumulada de cada caminho encontra-se em negrito à direita de cada nó na Figura 4.4.

Métricas sublinhadas representam métricas de caminhos sobreviventes.

Métricas em itálico representam ramos que incidem no nó “por cima” e métricas em não-itálico representam ramos que incidem no nó “por baixo” , já que, no máximo 2 ramos incidem em um nó para este decodificador.

A decodificação final é iniciada a partir do caminho sobrevivente de menor métrica acumulada, identificando cada ramo sobrevivente da direita para a esquerda na treliça, conforma mostra a Figura 4.5.

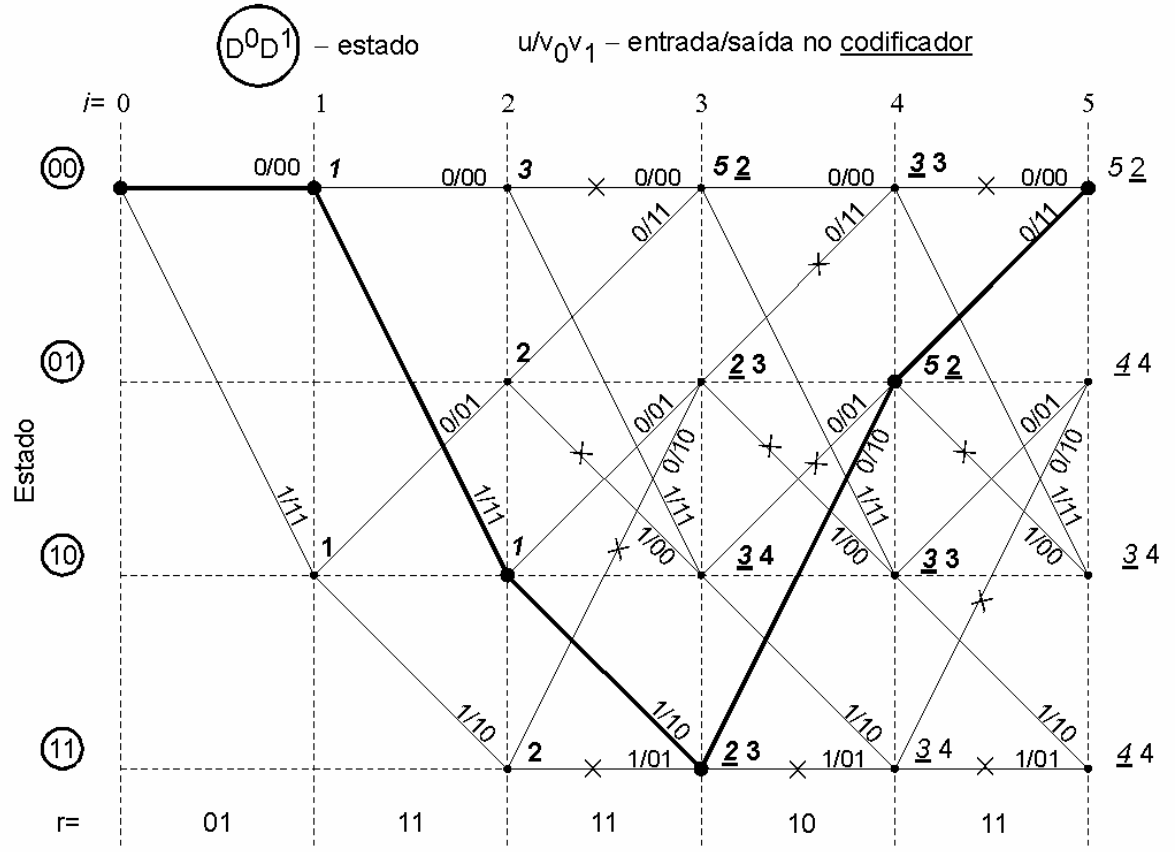


Figura 4.5: Decodificando a seqüência r da Tabela 4.8.

Ao lermos o valor de u nos identificadores u/v_0v_1 de cada ramo sobrevivente na Figura 4.5, verificamos que a seqüência originalmente transmitida foi $u=[01100]$,o que concorda com u mostrado na Tabela 4.8.

Portanto, o decodificador identificou e corrigiu os 2 erros.